
*Journal of Advances in Science and Technology*

*Vol. III, Issue No. VI, August-2012, ISSN 2230-9659*

# DESIGN AND IMPLEMENTATION OF IMAGE PROCESSING ON EPGA

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

*Journal of Advances in Science and Technology*
*Vol. III, Issue No. VI, August-2012, ISSN 2230-9659*

# Design and Implementation of Image Processing on EPGA

**K. Seetharam**

Research Scholar, CMJ University, Shillong, Meghalaya

***Abstract – The criticality in handling the graphics data in an EPGA device has been addressed in this work. We have successfully shown how image data can be successfully converted to raw binary data and can be transferred as a file to the EPGA memory using the RS-232 link. Our design is very useful for those EPGA devices (low cost) which don't have a dedicated input graphics port but still there is a need to perform the processing of image data. Moreover, we have detailed out the EDK development platform for building the embedded applications in EPGA systems using Micro blaze soft-core processor. In this work we have also given a brief overview of creating programs using System C language. This work can work as a guide for the researchers and engineers who wish to build an embedded system for image processing applications. This paper presents the design and implementation of image processing applications on field programmable gate array (EPGA). To improve the implementation time, Xilinx AccelDSP, software for generating hardware description language (HDL) from a high-level MATLAB description has been used. Two EPGA-based architectures for image processing have been proposed: Color Space Conversion and Edge Detection. The designs were implemented on Spartan 3A DSP and Vertex 5 devices. Obtained results are discussed and compared with others architectures. Nowadays, image processing applications are frequently preferred in such fields as industrial automation, security, health, and traffic control in parallel with the developments in technology. The most important criterion for the applications used in these fields is to ensure the system to run at high speed and real time. Thus, EPGAs are commonly used in such applications. In this study, some of the basic real time images processing algorithms are implemented in an EPGA-based development kit.***

***Keywords: EPGA, Image Processing, Real-Time Systems, Design Flow, Image Processing.***

----------♦----------

## INTRODUCTION

Nowadays, the importance of image processing is rapidly increasing in such fields as industrial automation, security, health, and traffic control in parallel with the developments in technology. The most challenging difficulty in applications used in these fields is to make system run real-time (Russ, 2011). It is not always possible to make the system run real-time by the use of software used on a general purpose computer since the resources of memory, CPU and peripheral devices in computers are limited. In most image processing applications, dozens of operations are performed on each pixel. That these operations are performed by general purpose processors sequentially leads to negative consequences in terms of both resource consumption and performance (Samarawickrama, *et. al.,* 2010). However, EPGAs has the capability to operate in a parallel way in terms of hardware, which distinguishes them from traditional processors. In this way, operations are divided into pieces in EPGAs and multiple operations can be done simultaneously. Image and video processing are an ever expanding and dynamic areas with applications reaching out into our everyday life such as in medicine, astronomy, ultrasonic imaging, remote sensing, space exploration, surveillance, authentication, automated industry inspection and in many more areas (Russ, 2011). Reconfigurable hardware in the form of (EPGAs) offers many performance and implementation benefits for executing video processing applications. EPGAs generally consist of logical blocks and some amount of Random Access Memory (RAM), all of which are wired by a vast array of interconnects. All logic in EPGA can be rewired, or reconfigured with different purposes as many times as a designer likes. One of the benefits of EPGA is its ability to execute operations in parallel, resulting in remarkable improvement in efficiency. The main advantage of EPGA-based design is the flexibility to exploit the inherently parallel nature of many image processing problems (Samarawickrama, *et. al.,* 2010).

## REVIEW OF LITERATURE

The difficulty of generating a design from a set of requirements and specifications increases as the

system becomes complex. These difficulties led to the development of electronic system level (ESL) design and verification (Moertti, 2002). which is an algorithm modeling methodology that focuses on a higher abstraction level using high-level languages such as C, C++, or MATLAB to model the entire behavior of the system with no initial link to its implementation. The ESL design and verification enables embedded system design, verification, and debugging for designing hardware and software implementation of custom system-on-EPGA (Akpan, 2012). The Xilinx AccelDSP tool (Ahmad, *et. al.,* 2010) is an advanced ESL design tool which transforms a MATLAB floating-point design into a hardware module that can be implemented in EPGA. The AccelDSP Synthesis Tool features an easy-to-use Graphical User Interface that controls an integrated environment with other design tools such as MATLAB, Xilinx ISE tools, and other industry standard HDL simulators and logic synthesizers. This paper presents the design and implementation of EPGA-based architecture for image processing by employing Xilinx AccelDSP tool. This tool has been selected, since it can converts automatically from high-level languages (HLLs) to register transfer level (RTL) HDL and even directly to EPGA configuration bit stream (Yu, *et. al.,* 2008).

## DESIGN FLOW FOR IMPLEMENTATION ON EPGA

The integration of Simulink and MATLAB from The Math Works (Sima, *et. al.,* 2003) and the EPGA design suite of tools (Jack, 2007) now allow embedded system development from a model-based view point which targets an EPGA. The AccelDSP software (Ahmad, *et. al.,* 2010) is the Mat lab signal processing model synthesis tool from Xilinx, which allows an algorithm developer to transform a Mat lab floating-point design into a hardware module that can be implemented in silicon. Its most interesting feature is that a synthesizable RTL HDL model and a Test bench can be achieved to ensure bit-true, cycle-accurate design verification. The tool also provides scripts that invoke and control downstream tools such as HDL simulators, RTL logic synthesizers and implementation tools. Three AccelDSP implementation options (flows) are available as illustrated in “Fig.1”. The default synthesis flow is called the ISE Synthesis Flow where the main objective is to create an implementation using ISE software and verify the design using HDL gate-level simulation (Jack, 2007. ITU-R, 2000. Saidani, *et. al.,* 2009). The second flow is called the System Generator flow. In this flow, an IP core is created for exporting and integrating with a larger System Generator design. The third flow, HW Co-Sim, is similar to the ISE flow but the objective is to simulate the design in hardware platform like a Virtex-4, a Virtex-5, or a Spartan-3A DSP Platform. Not only does the simulation run much faster, but this flow proves that the design will run in the target hardware. The AccelDSP IP Core Generators provide a direct path to hardware implementation for complex MATLAB built-in and toolbox functions, which when used with the AccelDSP synthesis tool, produces synthesizable and pre-verified intellectual property (IP) cores that enables and facilitate algorithmic synthesis for EPGAs (Akpan, 2012).

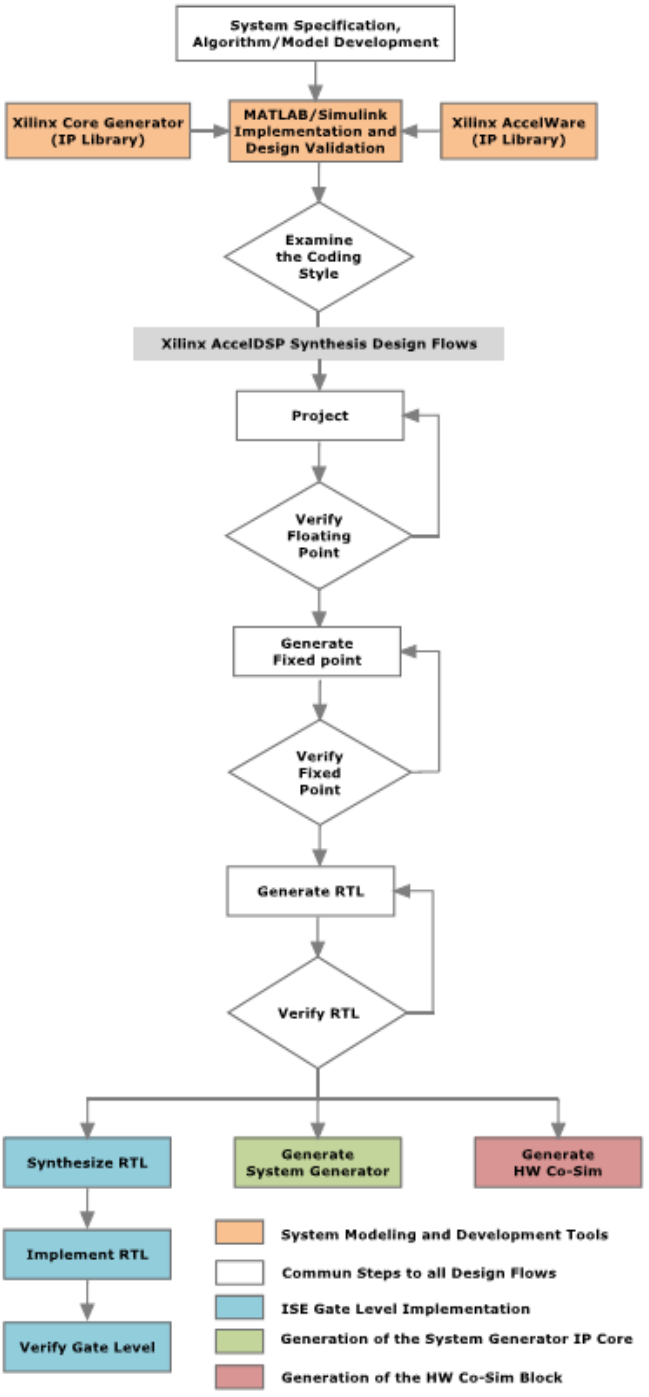


**Fig.1. From system specification and algorithm/model development to EPGA synthesis design flow options implementations**

## THE EPGA CARD USED

The Zed Board Zynq-7000 EPGA development kit (Jack, 2007) was used in this study. Zynq-7000 is different from standard EPGAs in that it contains Artix-7 EPGA and ARM Cortex-A9 processor on the same chip together. In EPGA card having this

system, those parts containing intense computations are performed on EPGA and the control parts not containing computations can be done on the processor by using software. The internal structure of Zynq is shown in Fig. 2. Zynq consists of two parts: Processing System and Programmable Logic. Processing System (PS) works like a traditional processor since it contains structures such as ARM Cortex-A9, Floating Point Unit, Memory Controller, Gigabit Ethernet Controller and USB Controller. Programmable Logic part, on the other hand, contains all the structures of a standard EPGA (Sapkal, *et. al.,* 2008. Canny, 1986. Behera, *et. al.,* 2012. Abbasi and Abbasi). The communication between PS and PL parts is provided by high performance data-paths.

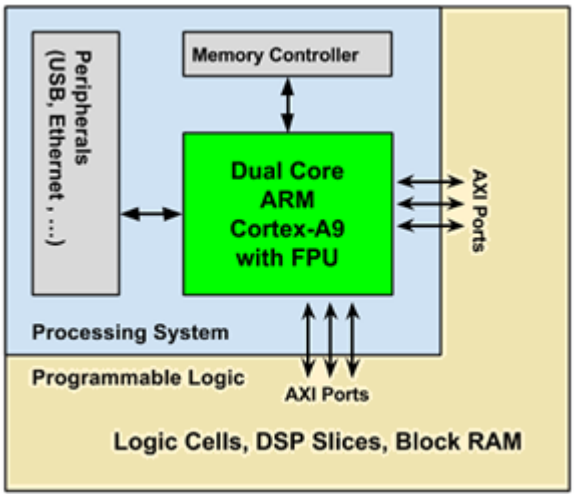


**Fig. 2 the internal structure of Zynq**

**General System**

The developed system consists of two parts: hardware and software. The software performs the operation of image capturing from the video source and transferring it to the hardware part. Any image source that can run on Linux (such as USB camera, network video stream, video file etc.) can be used in the software. In the hardware part, image processing algorithms are implemented. The schema of the general system formed by the co-use of software and hardware parts is shown in Fig. 3.

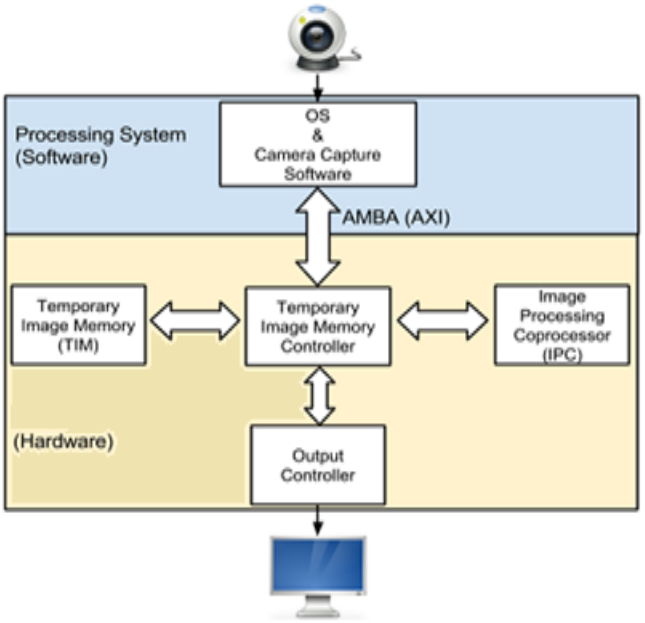


**Fig. 3 The schema of the general system**

In the system, video frame is firstly obtained from the video source via software and operating system. Then, the software transfers the parts to be processed on the video frame to the temporary image memory on the EPGA. The transfer of operation, on the hardware part, is controlled by Temporary Image Memory Controller (TIMC) module. The video is transferred to Temporary Image Memory (TIM) module via the TIMC. The processed and unprocessed forms of the image can be stored in the TIM module. Image Processing Co-processor (IPC) module performs image processing algorithms. The IPC module performs the image processing operations by fetching the pixels from TIM module and storing the processed forms of the pixels. These modules are explained in the 4th part. After the operations on the image are completed on the hardware part, the status notification is transmitted to the software. After this step, the processed image can be read via the software part if needed. The original and processed forms of the image are shown on the monitor by using the HDMI output of EPGA kit.

## IMAGE PROCESSING APPLICATIONS DEVELOPED WITH ACCELDSP

Two image processing applications have been designed and developed using Xilinx AccelDSP. A Color space conversion RGB to YCbCr and Sobel edge detector have been designed and implemented on EPGA.

**Color Space Conversion: RGB TO YCBCR**

A color space is a mathematical representation of a set of colors. The three most popular color models

are RGB (used in computer graphics); YIQ, YUV, or YCbCr (used in video systems) and CMYK (used in color printing).However, none of these color spaces are directly related to the intuitive notions of hue, saturation, and brightness. All of the color spaces can be derived from the RGB information supplied by devices such as cameras and scanners (Saidani, *et. al.,* 2009). "Fig.4" shows the basic steps in the AccelDSP Synthesis Flow (System Generator implementation option) with output results for the floating point and fixed point model respectively and the System Generator CSC IP Core generated.

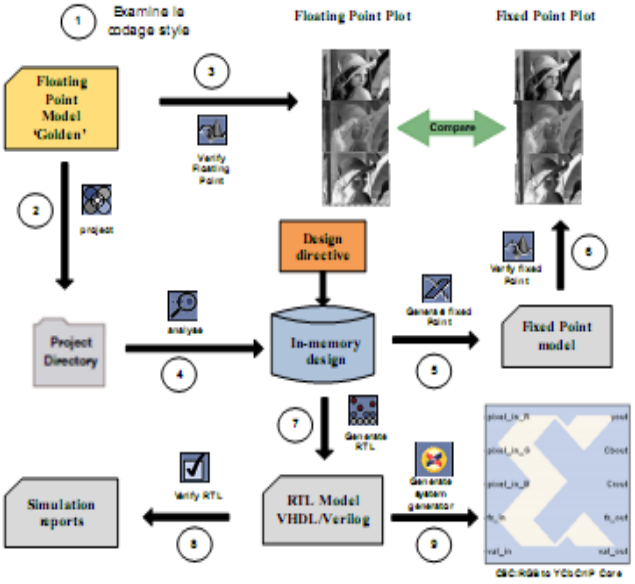


**Fig.4. the basic steps in the AccelDSP Synthesis Flow with output results for the CSC**

The IP Core block generated is exported and integrated with a larger System Generator design for hardware Co-simulation and implementation. "Fig.5" shows the design that uses the generated IP Core module and Xilinx block sets for RGB to YCbCr conversion. The hardware Co-simulation results for the CSC design for the input image are shown in "Fig.4"

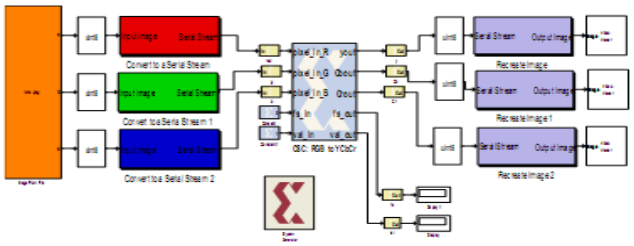

**Fig.5. the Design Model for RGB to YCbCr in MATLABSimulink/Xilinx System Generator**

## METHODOLOGY

From the development of EPGA technology, the methodology challenges the update of various EDA tools. Based on the standard development flow, initial efforts have been transferred to high-level design and synthesis. There are many conversion tools such as C-to-EPGA, State flow diagram to VHDL and Simulink/Mat lab-to-EPGA. The features of Xilinx AccelDSP-to-EPGA (Ahmad, *et. al.,* 2010) flow can be discussed as follows.

- Fast time-to-market for computer vision algorithms development. It could be described as a timely, advantageous option for developing in a much more comfortable way than that permitted by VHDL or Verilog hardware description languages (HDLs).

- Friendly graphical user interface (GUI) that features a Design Flow Manager to guide the designer quickly through the design transformation steps. The GUI also features a Project Explorer window that lets the designer graphically browse the design hierarchy and view the M-files and the generated HDL source files.

**Table 1 EPGA Resources Used In the Implementation for the CSC**

| | Spartan 3A DSP 3400 | | | Virtex 5 xc5vlx50-1ff676 | | |
|---|---|---|---|---|---|---|
| | Used | Available | % | Used | Available | % |
| Number of Slice Registers | 255 | 23872 | 1% | 114 | 28800 | 0 % |
| Number of Slice LUTs | 464 | 47744 | 0% | 393 | 28800 | 1 % |
| Number of LUT-FF pairs | 128 | 47744 | 0% | 111 | 396 | 28% |
| Number of bonded IOBs | 75 | 469 | 16 % | 75 | 440 | 17 % |
| Maximum Frequency | 53.4 MHz | | | 100.4 MHz | | |

**Table 2 EPGA Resources Used In the Implementation for the Sobel Edge Detector**

| | Spartan 3A DSP 3400 | | | Virtex 5 xc5vlx50-1ff676 | | |
|---|---|---|---|---|---|---|
| | Used | Available | % | Used | Available | % |
| Number of Slice Registers | 812 | 23872 | 3% | 480 | 28800 | 1 % |
| Number of Slice LUTs | 760 | 47744 | 1% | 581 | 28800 | 2 % |
| Number of LUT-FF pairs | 456 | 47744 | 0% | 167 | 894 | 18 % |
| Number of bonded IOBs | 34 | 469 | 7 % | 34 | 440 | 7 % |
| Maximum Frequency | 50.8 MHz | | | 100.2 MHz | | |

**Table 3 Performance Comparisons**

| | Our Design | | | Design [18] | | |
|---|---|---|---|---|---|---|
| | Used | Available | % | Used | Available | % |
| Number of Slices | 177 | 768 | 23 % | 204 | 768 | 26 % |
| Number of Slice Flip Flop | 401 | 1536 | 26 % | 280 | 1536 | 18 % |
| Number of 4 input LUTs | 277 | 1536 | 18 % | 202 | 1536 | 13% |
| Number of bonded IOBs | 34 | 124 | 27 % | 81 | 124 | 65 % |
| Number of GCLKS | 1 | 8 | 12 % | 1 | 8 | 12 % |
| Maximum Frequency | 54.505 MHz | | | 134.756MHz | | |

- AccelDSP is capable of generating a System Generator Block that can be used in a larger design. With the assistance of specified DSP blocks for EPGA, a design in Xilinx System generator can greatly shorten the development cycle from algorithm to hardware.

An important attribute of our design using AccelDSP was that the block sets generated in AccelDSP for Xilinx System Generator, are reusable and can be neatly divided into appropriate libraries each containing blocks specific to a certain field such as (for example) Image Processing Library (Gribbon, *et. al.,* 2006). The EPGA design made using high-level synthesis (HLS) tool needed much less effort than the

equivalent application implementation with traditional HDLs coding. One of the beneficial features of AccelDSP is its automated and flexible floating-to-fixed-point conversion.

## CONCLUSIONS

Implementation of a video processing algorithm on the EPGA is complex, tedious and error prone when using traditional design methodologies. Since time-to-market is very important, it is required to look at the product development cycle to reduce the design time and gain a competitive edge in the time to-market. Therefore, the adoption of high-level synthesis (HLS) tools is now getting into EPGA based designing. To ease the process of transforming a MATLAB floating point design into a hardware module, Xilinx introduced the AccelDSP software for rapid prototyping of an algorithm in MATLAB into hardware. In this paper, a Xilinx AccelDSP based approach is presented for image processing applications to minimize the time to market factor. A Color space conversion (CSC) RGB to YCbCr and Sobel edge detector have been designed and implemented on EPGA. The designs were implemented on Spartan 3A DSP and Vertex 5 devices and their utilization summaries are compared.

## REFERENCES

A. Ahmad, A. Amira, H. Rabah, Y. Berviller (2010). "EPGA-based Architectures of Finite Radon Transform for Medical Image De-noising", In IEEE APCCAS, pp. 20-23.

A. M. Sapkal, M. Munot, M. A. Joshi (January 2008). "R' G'B' to Y'CbCr Color Space Conversion Using EPGA", In IET International Conference on Wireless, Mobile and Multimedia Network 2008, Volume, Issue, 11-12, pp.255 – 258.

G. Moertti (February, 01 2002). "System-level design merits a closer look: the complexity of today's designs requires system-level", EDN Asia, pp. 22-28.

G. Yu, T. Vladimirova, X. Wu, M. N. Sweeting (2008). "A New High-Level Reconfigurable Lossless Image Compression System for Space Applications", In NASA/ESA Conference on Adaptive Hardware and Systems IEEE, pp. 183–190.

ITU-R (2000). BT.601-4, "Parameter Values for the HDTV Standards for Production and International Program Exchange", www.itu.int

J. C. Russ (2011). "The Image Processing Hand book", Sixth Edition, CRC Press.

J. Canny (June1986). "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell, vol. PAMI-8, no.6, pp. 679-698.

K. Jack (2007). "Video Demystified: A Handbook for the Digital Engineer", LLH Technology Publishing, Fifth Edition, 2007.

K. T. Gribbon, D. G. Bailey, and C. T. Johnston (January 2006). "Using design patterns to overcome image processing constraints on EPGAs," Third IEEE International Workshop on Electronic Design, Test and Applications DELTA, pp. 47– 56.

M. Samarawickrama, R. Rodrigo, and A. Pasqual (2010). "HLS Approach in Designing EPGABased Custom Coprocessor for Image Preprocessing", 5th international conference on ICIAF, IEEE, pp. 167 -171.

M. Sima, S. Vassiliadis, S. Cotofana and J. T.J. Van Eijndhoven (June 2003). "Color space conversion for MPEG decoding on EPGA-augmented trimedia processor", Proceedings IEEE International Conference on Application-Specific Systems, Architectures and Processors, pp. 250-259.

S. Behera, M. N. Mohanty, S. Patnaik (2012). "A Comparative Analysis on Edge Detection of Colloid Cyst: A Medical Imaging Approach," Soft Computing Techniques in Vision Science, Studies in Computational Intelligence, Springer, Volume 395, pp. 63-85.

T. A. Abbasi and M.U. Abbasi, "A proposed EPGA based architecture for sobel edge detection operator", J. of Active and Passive Electronic Devices, Vol. 2, pp. 271–277.

T. Saidani, D. Dia, W. Elhamzi, M. Atri and R. Tourki (2009). "Hardware Co-simulation for Video Processing Using Xilinx System Generator". Proceedings of the World Congress on Engineering 2009 Vol. I, WCE 2009, July 1 - 3, 2009, London, U.K.

V.A. Akpan (2012). "Model-Based EPGA EmbeddedProcessor Systems Design Methodologies: Modeling, Syntheses, Implementation and Validation". Afr J. of Comp & ICTs.Vol 5, No.1 pp. 1– 26.